SPARK AND GRAPHX BASED DE NOVO GENOME ASSEMBLER

Anas Abu-Doleh[1] and Ümit V. Çatalyürek[12]

[1]Dept. of Electrical and Computer Engineering [2]Dept. of Biomedical Informatics, Dept. of Computer Science & Engineering, College of Public Health-Division of Biostatistics, The Ohio State University

Introduction

The recent advancements in high-throughput genome sequencing technologies have accelerated the efficient discovery and extraction of various genomic sequences. A correct and cost-effective analysis of generated datasets raises several computational challenges.

De novo genome assembly is the process of stitching a set of short sequence reads to generate longer DNA sequences and it is used when there is no reference sequence. De novo genome assembly is a crucial step in many bioinformatics pipelines. However, there are challenges:

- Massive size of generated data: The expectation, in the next 10 years, the sequencing rate will reach one zetabase per year.
- Sequences similarity: The repeated regions make the assembly problem more complicated.
- Errors: Sequencing process, the variations in sequences of the subspecies (SNPs) and short insertion and deletion in the sequences require more processing.

Aim

We addressed the problem of parallelizing the de Bruijn graph based de novo genome assembly on distributed memory systems. Spaler, a new tool, which assembles short reads efficiently and accurately. Spaler is based on Spark framework and GraphX API.

Methods

Spaler consists of two main parts (Fig. 1): 1- De Bruijn graph (DBG) construction. 2- Contigs generating. DBG construction (Fig. 2) starts with processing each read and extracts the overlapped k+1-mers. The two complements of the DNA segment is represented by each k+1-mer by a single entity to reduce memory. A filter operation is applied to remove low coverage k+1-mers. Each k+1-mer represents an edge between two overlapped kmers that share k-1 base pairs. From the set of filtered k+1-mers, Spaler extracts the vertices and edges and constructs the graph using GraphX.

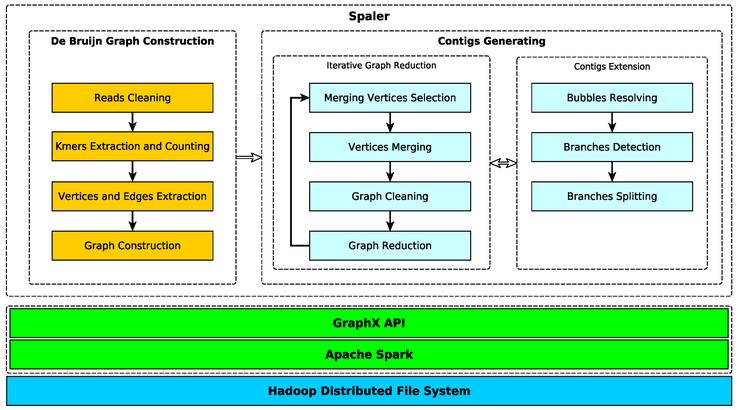


Figure 1: Spaler flowchart

Contigs generating part consists of two steps: 1- Iterative graph reduction, 2- contigs extension. Spaler splits the sequences paths into sub-paths of short length and generates a new DBG of reduced size by iteratively merging the vertices of these sub-paths. Contigs extension consists of bubbles resolving, and 3-degree and 4-degree branches resolving. After that, Spaler applies the iterative graph reduction to merge the newly generated vertices before generating the results.

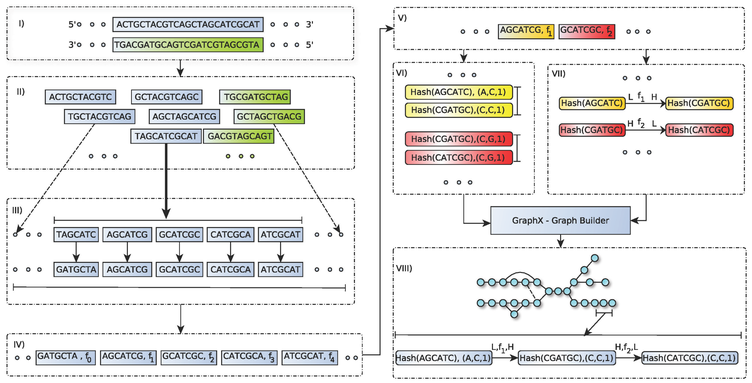


Figure 2: De Bruijn graph construction flowchart

Results

We compared Spaler with the state of the art distributed memory assemblers: Ray, AbySS and SWAP-Assembler (see Table 1 and Figs 3-4). We used a cluster of 32 nodes, each equipped with dual Intel Xeon E5520 Quad-core CPU and main memory of 48GB.

Table 1: Quality results of the human chromosome 14 assemblies

Assembly	Spaler	Ray	ABySS	SWAP
#Contigs	35,792	35,285	34,277	35,978
Largest contig	26,529	21,562	26,559	28,844
Total length	64,203	68,698	65,665	69,632
N50	2,410	2,707	2,663	2,693
#Misassemblies	67	12	6	2,766

Figure 3: For human chromosome 14, the execution times of using fixed parallelism level and using adaptive adjusting of the parallelism level for the two main stages of Spaler using 128 cores.

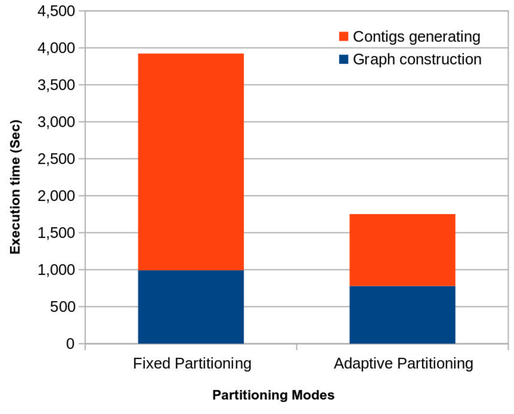
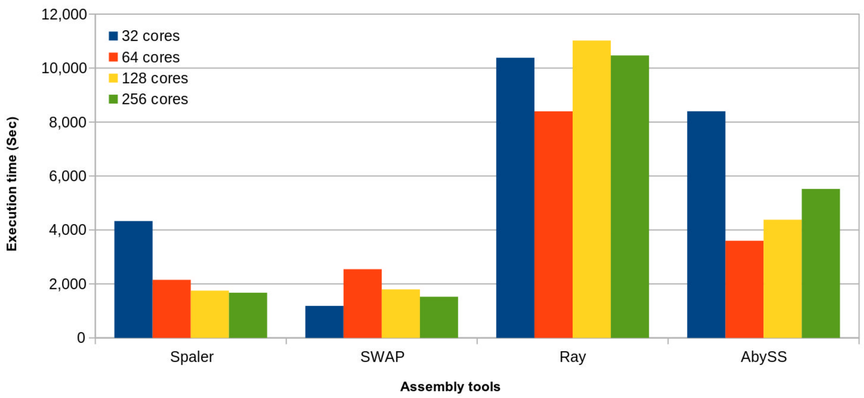


Figure 4: Execution time of human chromosome 14 assembly



Conclusion

Spaler is a Spark and GraphX based de novo genome assembler. Spaler uses an efficient algorithm based on an iterative graph reduction technique (Fig. 1). We showed the effects of partitioning size on the running time (Fig. 3). It proves that Spaler scales better than existing tools (Fig. 4) and produces comparable or better results in terms of solution quality (Table 1).